

인공위성 영상에서 객체 탐지 하기

SIA x AIFFEL 최종 해커톤 발표

오디조 : 김주현, 정우일, 서무일, 곽서연

Contents

01

프로젝트 팀 소개 및 역할

02

프로젝트 배경

03

프로젝트 수행 절차 및 방법

04

프로젝트 수행 과정

05

프로젝트 수행 결과

06

느낀 점

01.

프로젝트 팀 소개 및 역할

::1. 프로젝트 팀 구성 및 역할



팀장

김주현

- EDA 전/후 처리 담당
- 환경설정 담당



팀원

정우일

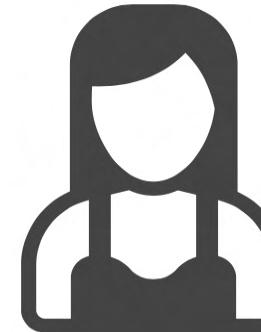
- 모델링 담당
- 자료 수집 담당
- 깃 허브



팀원

서무일

- EDA 전/후 처리 담당
- 환경설정 담당



팀원

곽서연

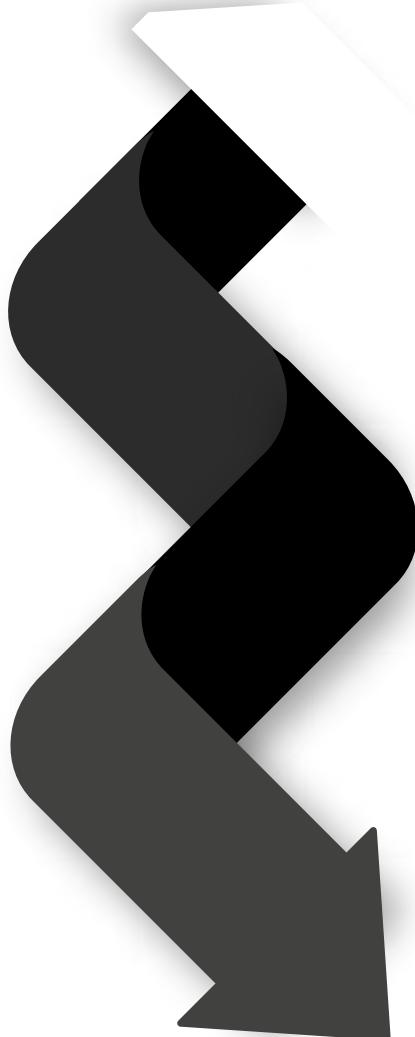
- 모델링 담당
- 자료 정리 담당
- 시각화 담당

02.

프로젝트 배경

2-1. AI Hub 데이터 셋 설명

::2. 프로젝트 목표



Step 01

위성 영상 지도에서 객체를 검출한다.

Step 02

씬 단위 위성 영상 지도에서 객체를 검출한다.

Step 03

모델 성능을 향상한다.

:: 2. 프로젝트 배경

- 프로젝트 주제
 - 아리랑 3A호(KOMPSAT-3A)에서 취득한 위성영상에서 객체 검출
 - 대형 위성영상(10,000x10,000 이상의 크기)에서 객체 검출
 - 정확도 등 모델 성능 향상
- AIFFEL 교육과정 내용과의 관련성
 - 일반적으로 측면에서 촬영된 사진의 객체 검출의 원리를 응용하여 위성 영상의 객체 검출을 수행
 - 기존 모델의 수정 및 이용, 시각화, 전/후 처리 등 AIFFEL 교육과정에서 배운 내용들을 종합하여 응용
- 개발환경
 - 주 플랫폼 : 구글 클라우드 플랫폼[GCP] (4 CPU , GPU : Tesla-v 100)
 - 부 플랫폼 : 구글 코랩, AIFFEL 클라우드
 - OS : Ubuntu 18.04
 - 사용 언어, 라이브러리 등등 : python 3.5+, pytorch 1.1, CUDA 9.0+ 등등

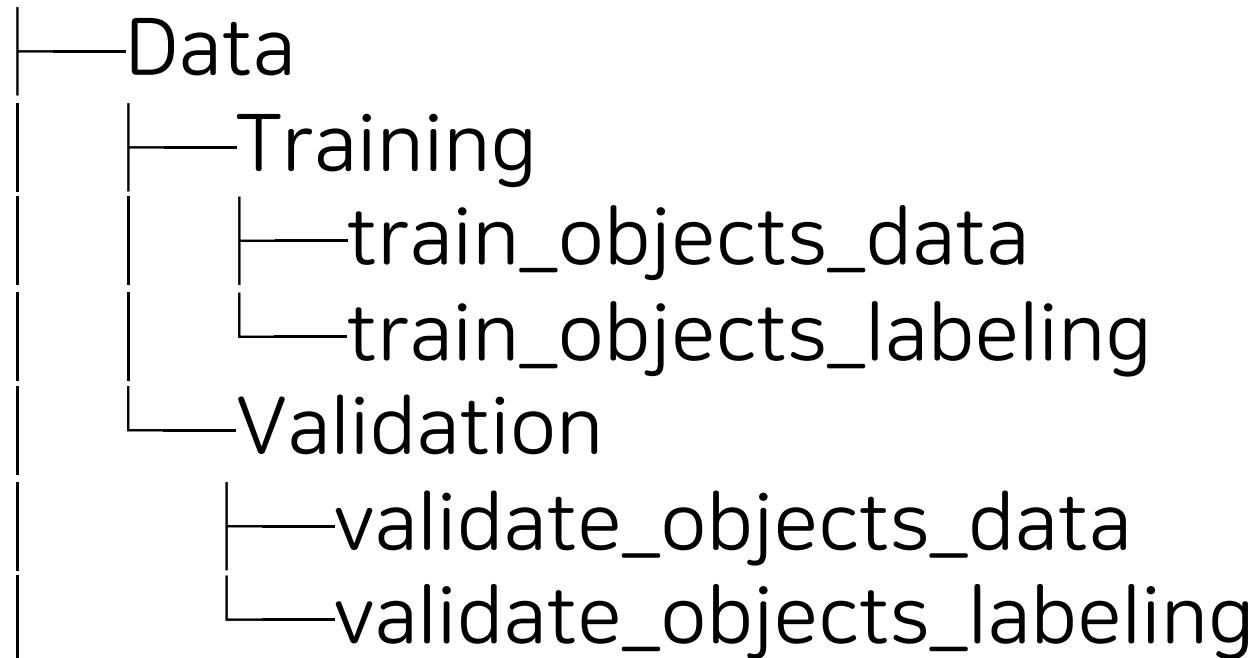
:: 2. 데이터셋 설명

데이터셋

- 데이터셋 (Image)
 - 아리랑 3A호(KOMPSAT-3A)에서 취득한 위성영상 패치
 - > Scene 형태의 영상을 AI를 위해서 1024 크기의 패치로 자른 형태
 - R,G,B 3채널로 생성된 영상 활용
 - GSD(Ground Sample Distance): 0.55m
 - 15종의 다양한 객체들 (소형승용차, 트럭, 버스, 기차, 운동경기장 등)
- Label
 - 객체의 길이와 방향을 알 수 있는 Rotated Bounding Box 형태로 구축 (geojson 파일)

:: 2. 데이터 셋 설명

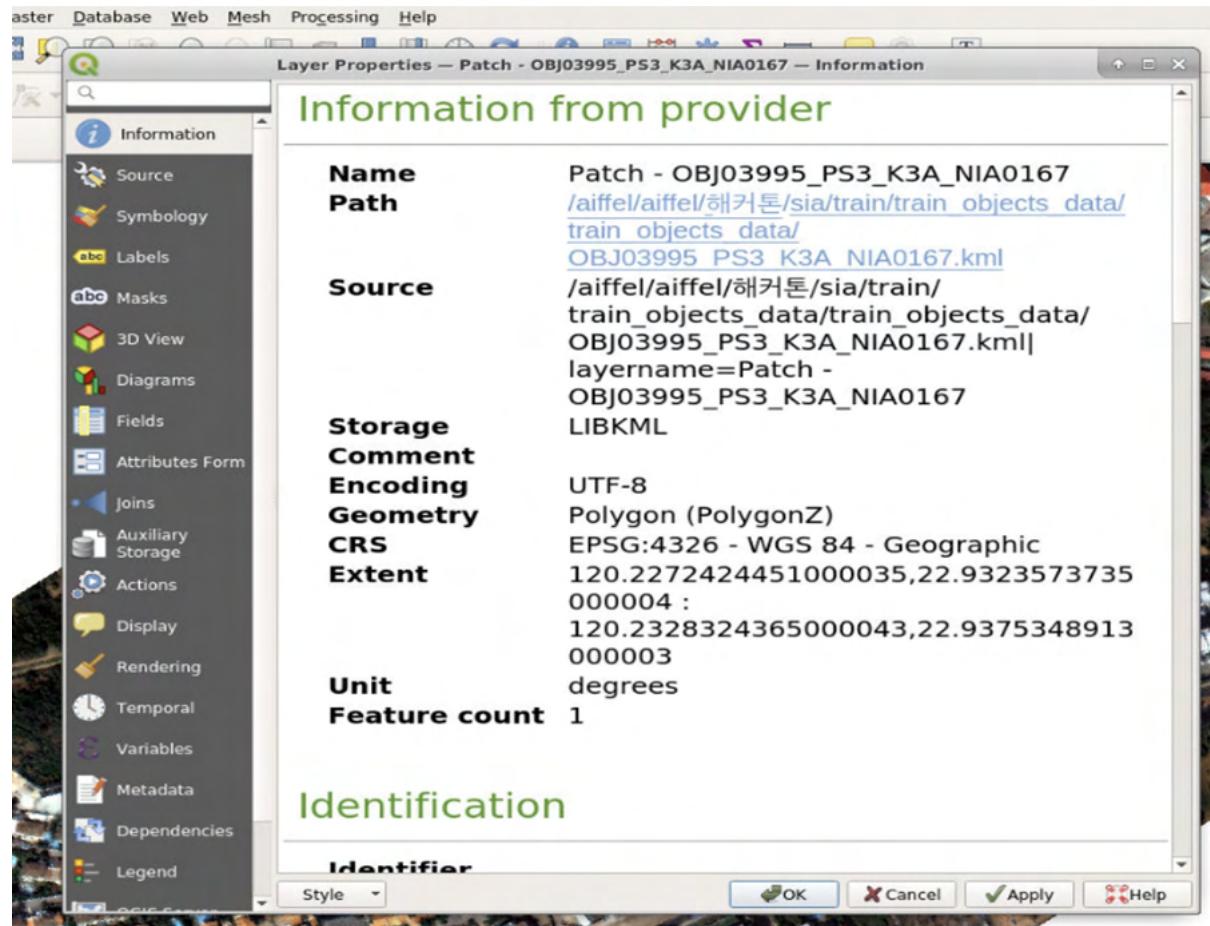
데이터 구성



Data의 폴더는 위와 같은 디렉토리 트리 구조를 가짐.

:: 2. 데이터셋 설명

- kml - 키홀 마크업 언어이며, 연계되는 png 파일의 실제 좌표(위, 경도)를 포함



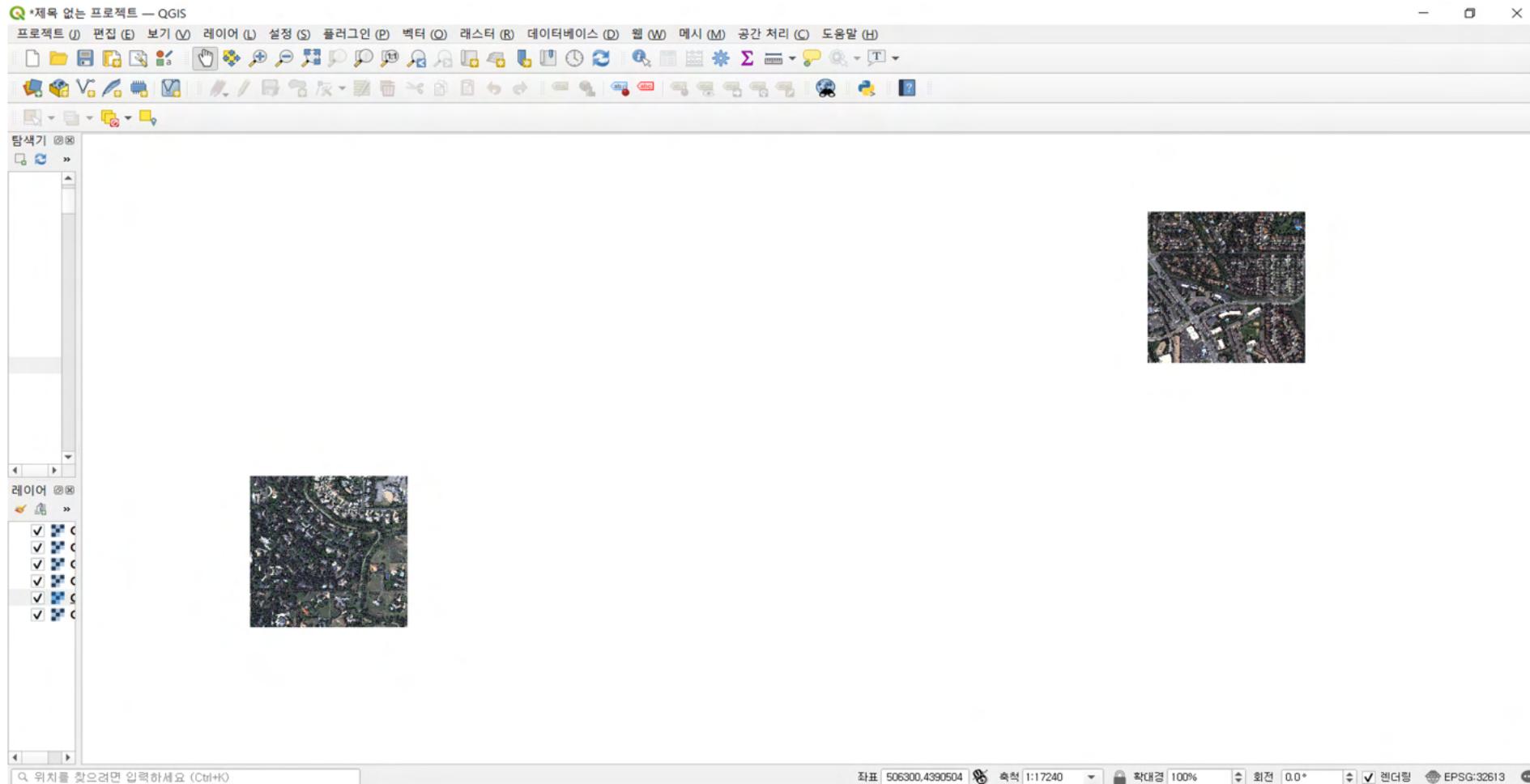
:: 2. 데이터셋 설명

- png - 일반적으로 사용되는 영상 이미지 확장자



:: 2. 데이터셋 설명

tif - 지리 정보를 포함한 영상 데이터, QGIS 프로그램을 통해 열어볼 수 있음



:: 2. 데이터셋 설명

● objects data에 대응하는 json파일 존재

```
{"features" : list 형태로 하나의 원소가 하나의 객체를 의미하며 각각은 아래와 같은 구조를 가짐. }

{"geometry" : {"coordinates" : [[x1, y1, z1],[x2, y2, z2],[x3, y3, z3],[x4, y4, z4]], #이미지 patch의 실제 지리 좌표
               "type" : "Polygon"},

"properties" : {"object_imcoords" : "x1, y1, x2, y2, x3, y3, x4, y4", # float 형태, 좌 상단에서 시작해서 시계 방향순
                "image_id" : 해당 이미지 파일 이름,
                "ingest_time": 작업 날짜,
                "type_id" : 객체 클래스 id,
                "type_name" : 객체 클래스 이름}}
```

:: 2. 데이터셋 설명

objects data에 대응하는 json파일 존재

```
'ingest_time': '2020-11-20T07:19:42.862436Z',
'object_angle': 6.283185307179586,
'object_imcoords': '560.0341362959969,874.8681059038561,649.2163280355734,874.8681059038561,649.2163280355733,954.855945898074,560.0341362959967,954.855945898074',
'road_imcoords': 'EMPTY',
'type_id': '16',
'type_name': 'roundabout'},
'type': 'Feature'},
```

- **object_angle**
 - 물체가 회전한 각도를 의미하며 수직선 기준으로 시계방향으로 얼마나 회전 했는지를 의미
- **object_imcoords**
 - 총 8개의 데이터가 주어져 있고 두 개씩 x, y쌍으로 4개의 좌표를 말해주는 데이터
 - 좌표 순서는 left_top -> right_top -> right_bottom -> left_bottom 순서

:: 2. 데이터셋 설명

● type_id, type_name

Object Class 수 : 21개

 Small ship	 Truck	 Indoor playground	 Grouped container
 Large ship	 Train	 Outdoor playground	 Swimming pool
 Civilian aircraft	 Crane	 Helipad	 Etc.
 Military aircraft	 Bridge	 Roundabout	
 Small car	 Oil tank	 Helicopter	
 Bus	 Dam	 Individual container	

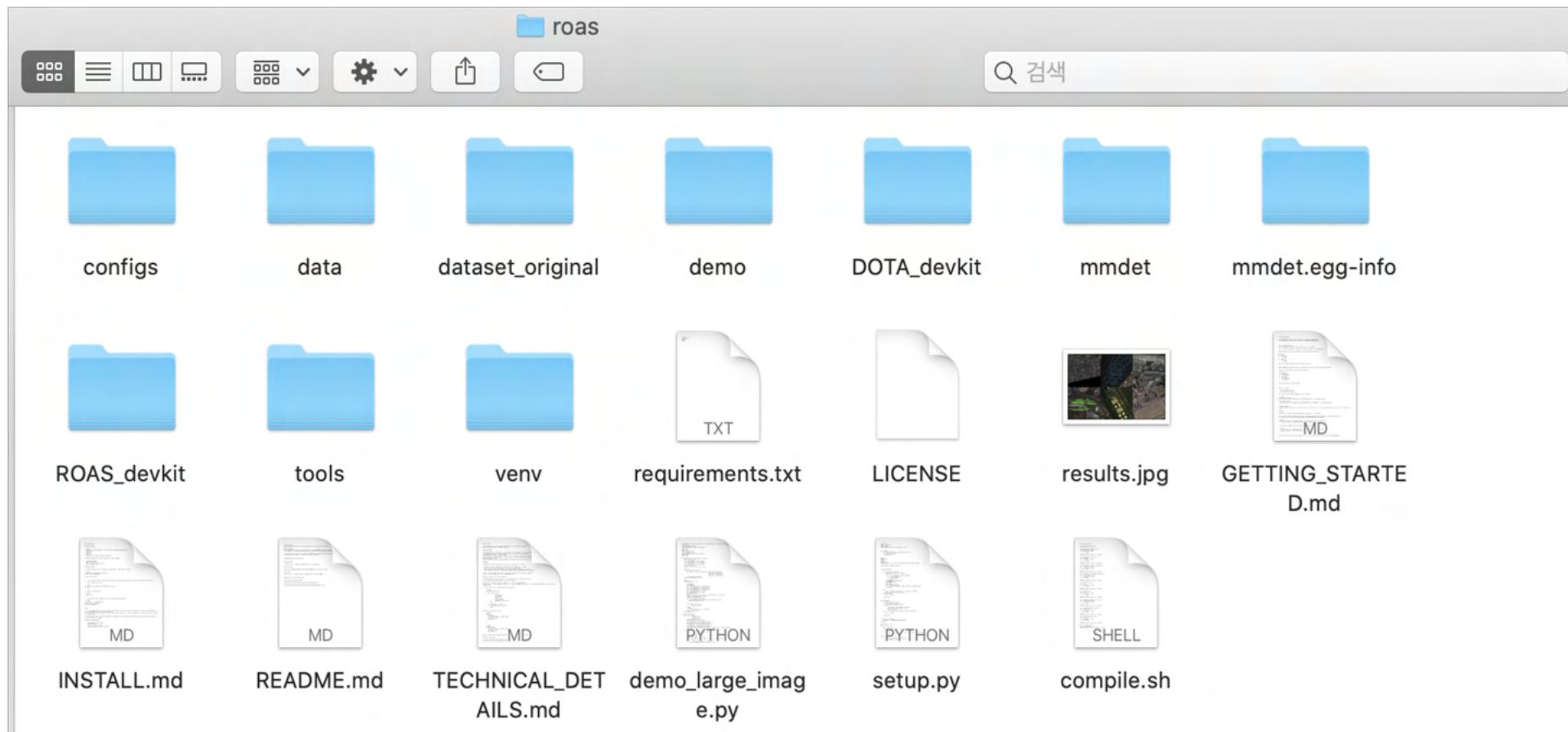
:: 2. 데이터셋 설명



bbox의 좌표를 png파일에 line으로 이어주면 위와 같이 object bbox가 plot되는 것을 볼 수 있습니다.

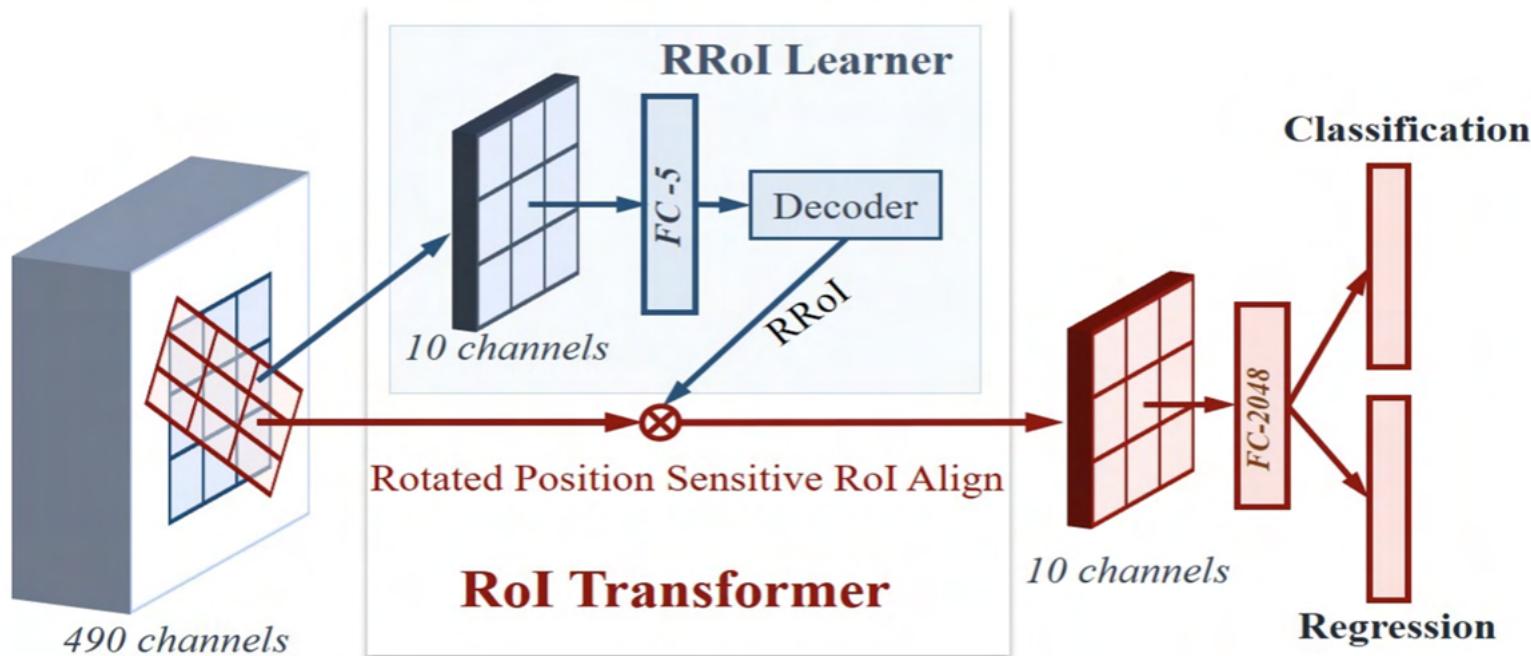
::2. 데이터셋 설명

참조 모델



::: 2. 데이터셋 설명

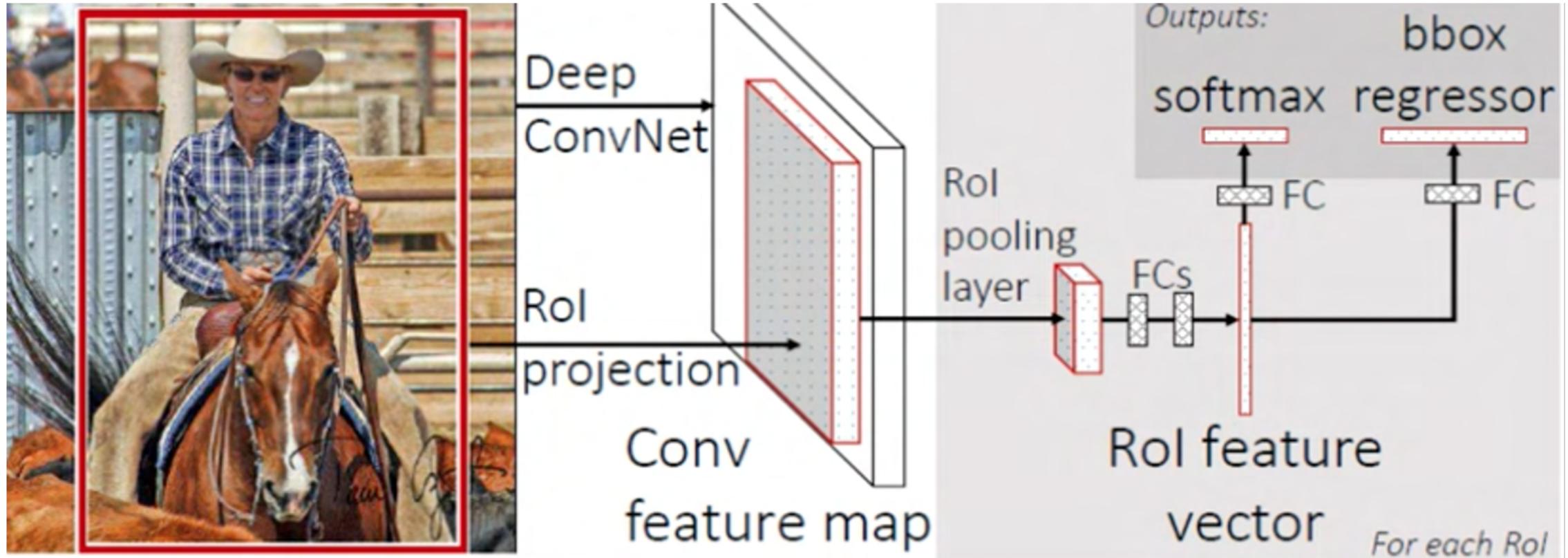
SIA 제공 모델



- Baseline: CVPR2019에서 소개된 ROI Transformer+Faster R-CNN
- Backbone: ResNet101 model
- Framework: DOTA 연구팀이 공개한 Benchmark 구조를 활용하여 재구성

::: 3. 프로젝트 수행 절차 및 방법

● SIA 제공 모델

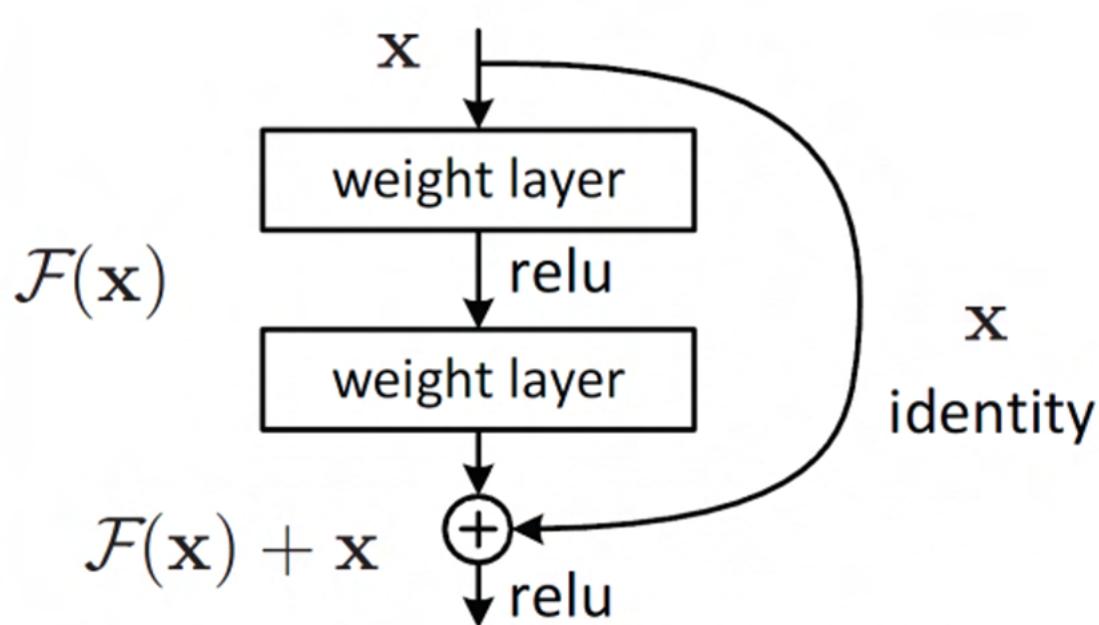


::4. 프로젝트 수행 과정

● backbone

torchvision:resnet101을 이용함

101층 깊이의 합성곱 신경망입니다. ImageNet 데이터베이스에서
백만 개 이상의 이미지에 대해 훈련된 네트워크의 사전 훈련된 버전을 로드



::4. 프로젝트 수행 과정

● backbone

FPN(Feature Pyramid Network) 을 이용함

FPN은 객체 탐지를 위한 네트워크로, COCO 2016 및 2015 챌린지에서 우승하기도 했습니다.

FPN에서는 Feature Map의 피라미드를 생성합니다. RPN(이전 섹션에서 설명)을 적용하여 ROI를 생성합니다. ROI의 크기를 기반으로, 우리는 특징 패치를 추출하기 위해 가장 적절한 스케일의 특징 맵 레이어를 선택합니다.

method	backbone	competition	image pyramid	test-dev					test-std				
				AP _{@.5}	AP	AP _s	AP _m	AP _l	AP _{@.5}	AP	AP _s	AP _m	AP _l
ours, Faster R-CNN on FPN	ResNet-101	-		59.1	36.2	18.2	39.0	48.2	58.5	35.8	17.5	38.7	47.8
<i>Competition-winning single-model results follow:</i>													
G-RMI [†]	Inception-ResNet	2016	-	34.7	-	-	-	-	-	-	-	-	-
AttractioNet [‡] [10]	VGG16 + Wide ResNet [§]	2016	✓	53.4	35.7	15.6	38.0	52.7	52.9	35.3	14.7	37.6	51.9
Faster R-CNN +++ [16]	ResNet-101	2015	✓	55.7	34.9	15.6	38.7	50.9	-	-	-	-	-
Multipath [40] (on minival)	VGG-16	2015	-	49.6	31.5	-	-	-	-	-	-	-	-
ION [‡] [2]	VGG-16	2015	-	53.4	31.2	12.8	32.9	45.2	52.9	30.7	11.8	32.8	44.8

03.

프로젝트 수행 절차 및 방법

프로젝트 수행 절차

11/05~11/09

11/10~11/17

11/18~11/24

11/25~12/01

12/02~12/14

1. aihub 데이터셋
EDA

2. 모델 선정
라이브러리 논문 탐색

3. 모델 선정
GroundTruth 확인
GCP 환경설정
(map)평가지표 확인

4. gdal 활용방안 모색
대용량 이미지 검출

5. 대형 이미지
crop 및 merge
검출 성능 향상 시도

3. 프로젝트 실행 절차 및 방법

AI Hub에서 제공된 baseline(model zoo 포함)을
실험해 보기 위해서 GCP에서 환경 구축 및 설정

The screenshot shows the Google Cloud Platform Compute Engine VM Instances page. The URL is `console.cloud.google.com/compute/instances?project=aiffel-ic-4&cloudshell=false`. The left sidebar shows 'Compute Engine' under '가상 머신'. The main area displays a table of instances:

상태	이름	영역	권장사항	다음에	연결
<input type="checkbox"/>	aiffel-ic-4	us-west1-b			SSH

On the right, there is an 'INSTANCES SELECT' panel with tabs for 'PERMISSIONS', 'LABELS', and 'MONITORING'. A message at the bottom says '리소스를 1개 이상 선택하세요.'

3. 프로젝트 수행 절차 및 방법

```
(star) totkfka4@aiffel-ic-4:~/roas$ python ROAS_devkit/geojson2csv.py
(star) totkfka4@aiffel-ic-4:~/roas$ python3 ROAS_devkit/evaluate_roas.py --gt_csv_path labels.csv --pred_csv_path result.csv
mAP: {'ap': array([      nan, 0.6599635 , 0.50300831, 0.67943976, 0.33848667,
       0.10783679, 0.06465538, 0.1601725 , 0.09963676, 0.18370656,
       0.1705546 , 0.84027423, 0.20220459, 0.          , 0.19450334,
       0.23933931, 0.69811641]), 'map': 0.32136866993785046}
(star) totkfka4@aiffel-ic-4:~/roas$
```

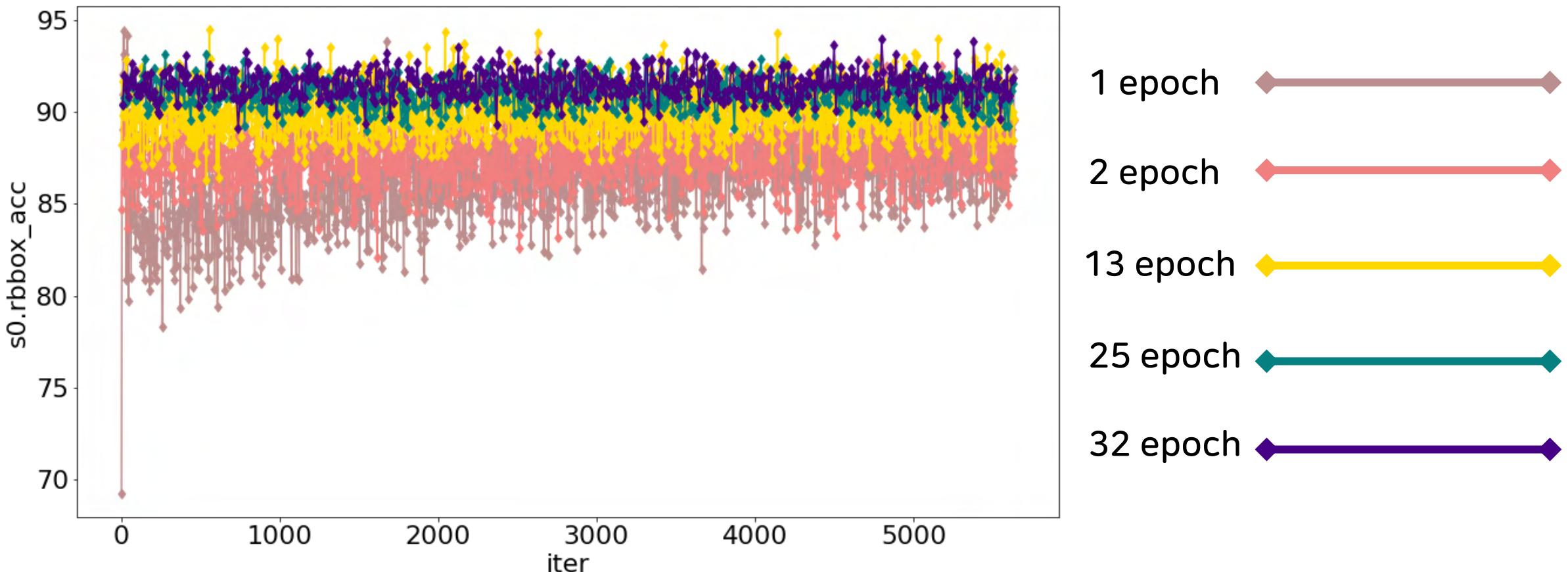
- Fast R-cnn의 Multi-task loss의 경우 bounding box regressing 을 통해서 실제 Ground Truth 값과 유사하게 학습이 되도록 유도합니다.
 - 배경일 경우, 바운딩 박스가 없기 때문에 0을 의미하고 0이 아닐 경우에는 그에 해당하는 Ground Truth에 맞게 학습이 되도록 Loss를 구성합니다.
- > Regression loss의 경우에는 더 smooth한 학습이 가능하도록 smooth (L1) 함수를 사용합니다.

04.

프로젝트 수행 과정

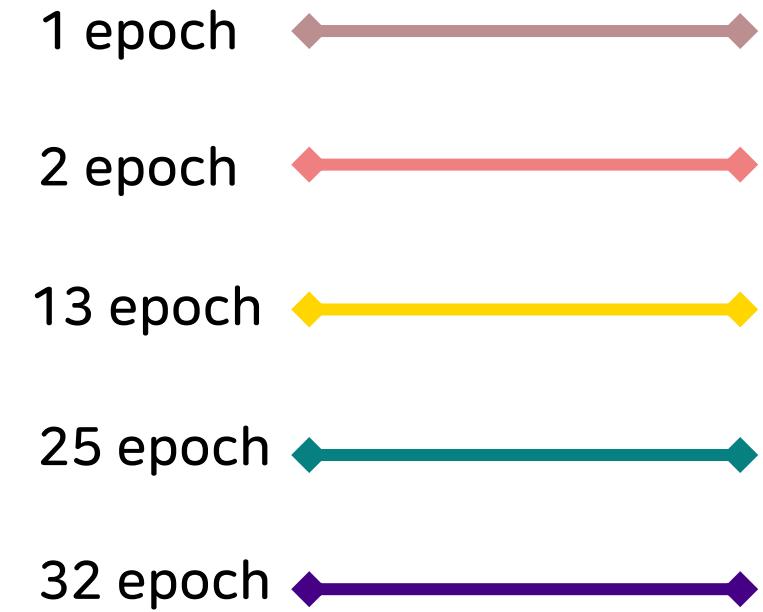
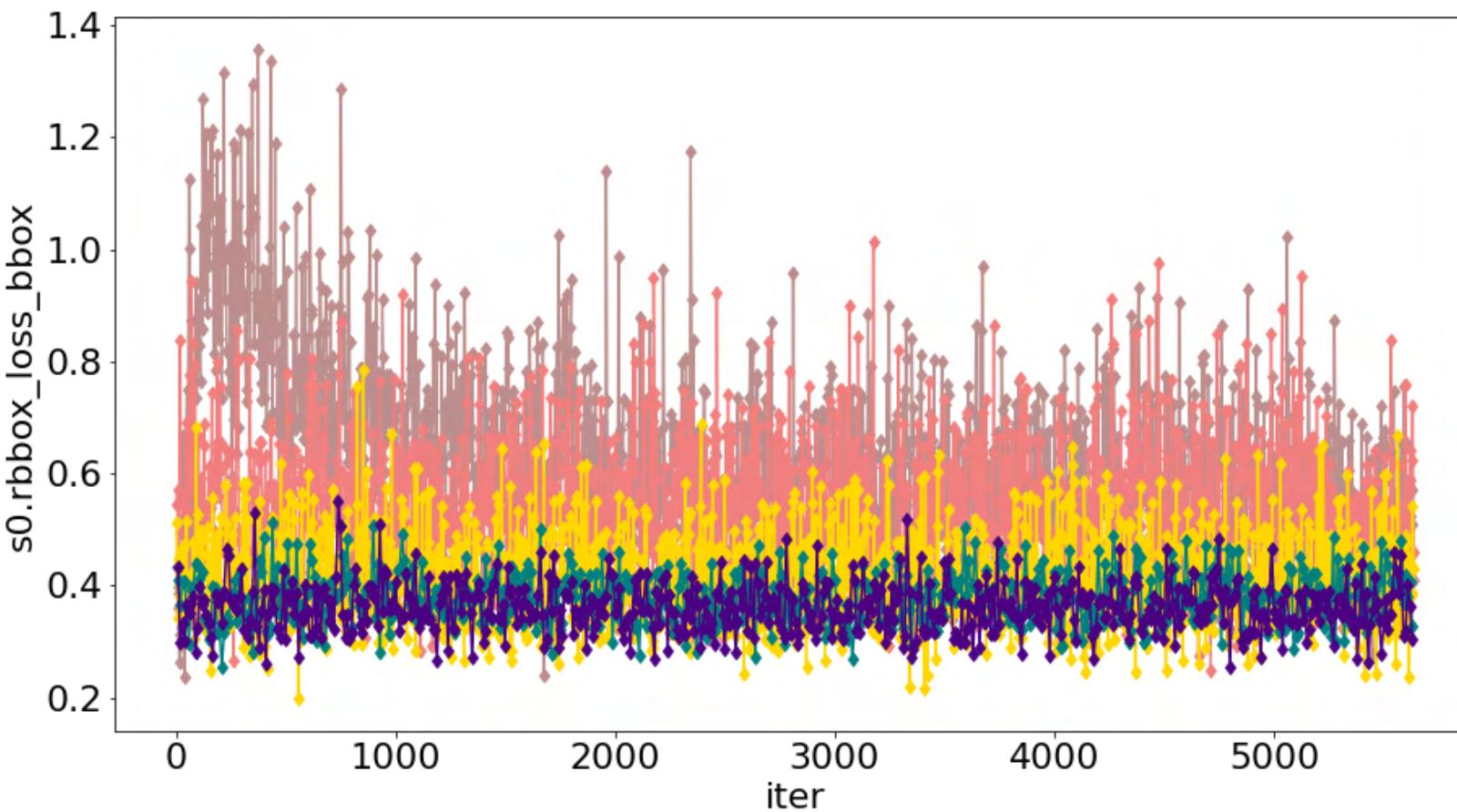
::5. 프로젝트 수행 결과

● rbbox_acc 시각화



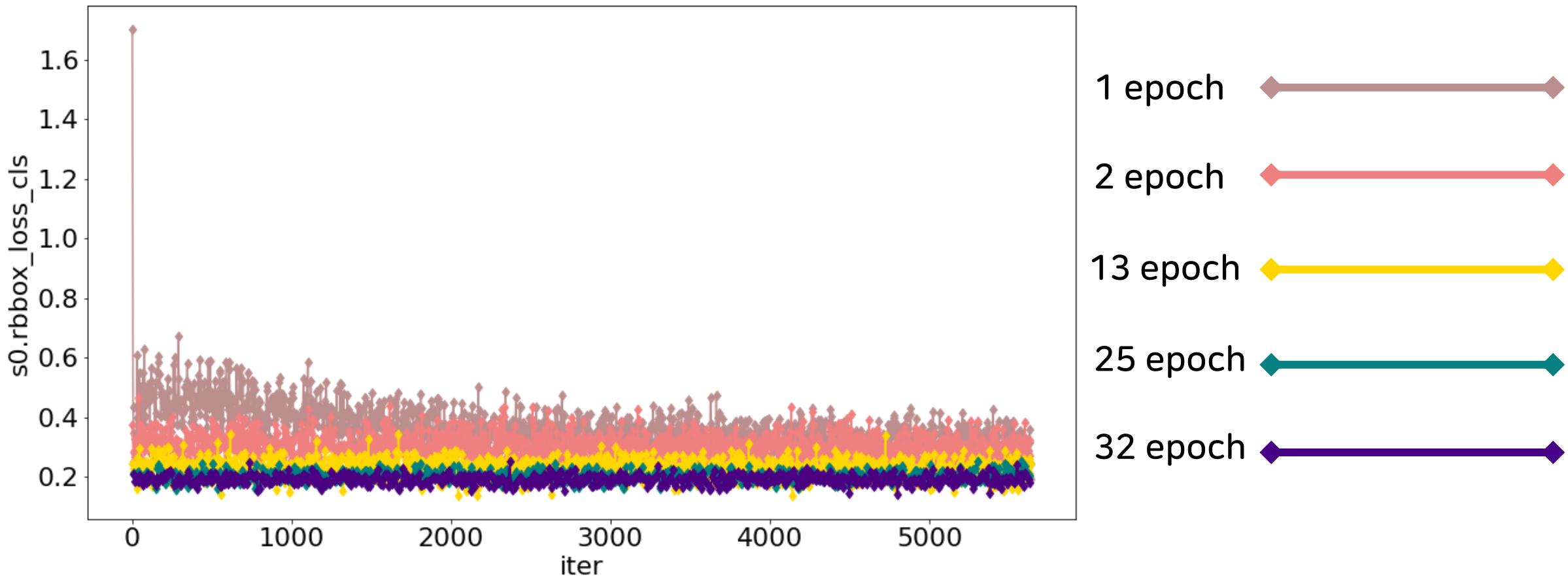
::5. 프로젝트 수행 결과

● rbbox_loss_bbox 시각화



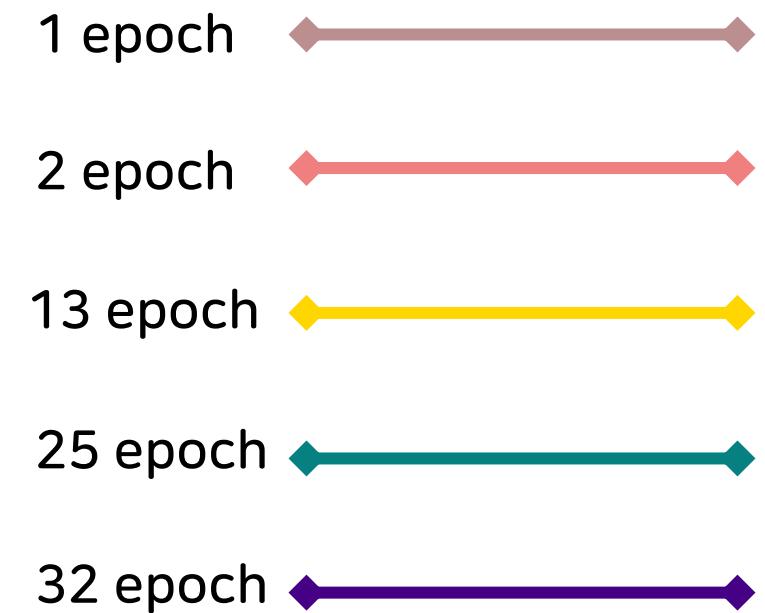
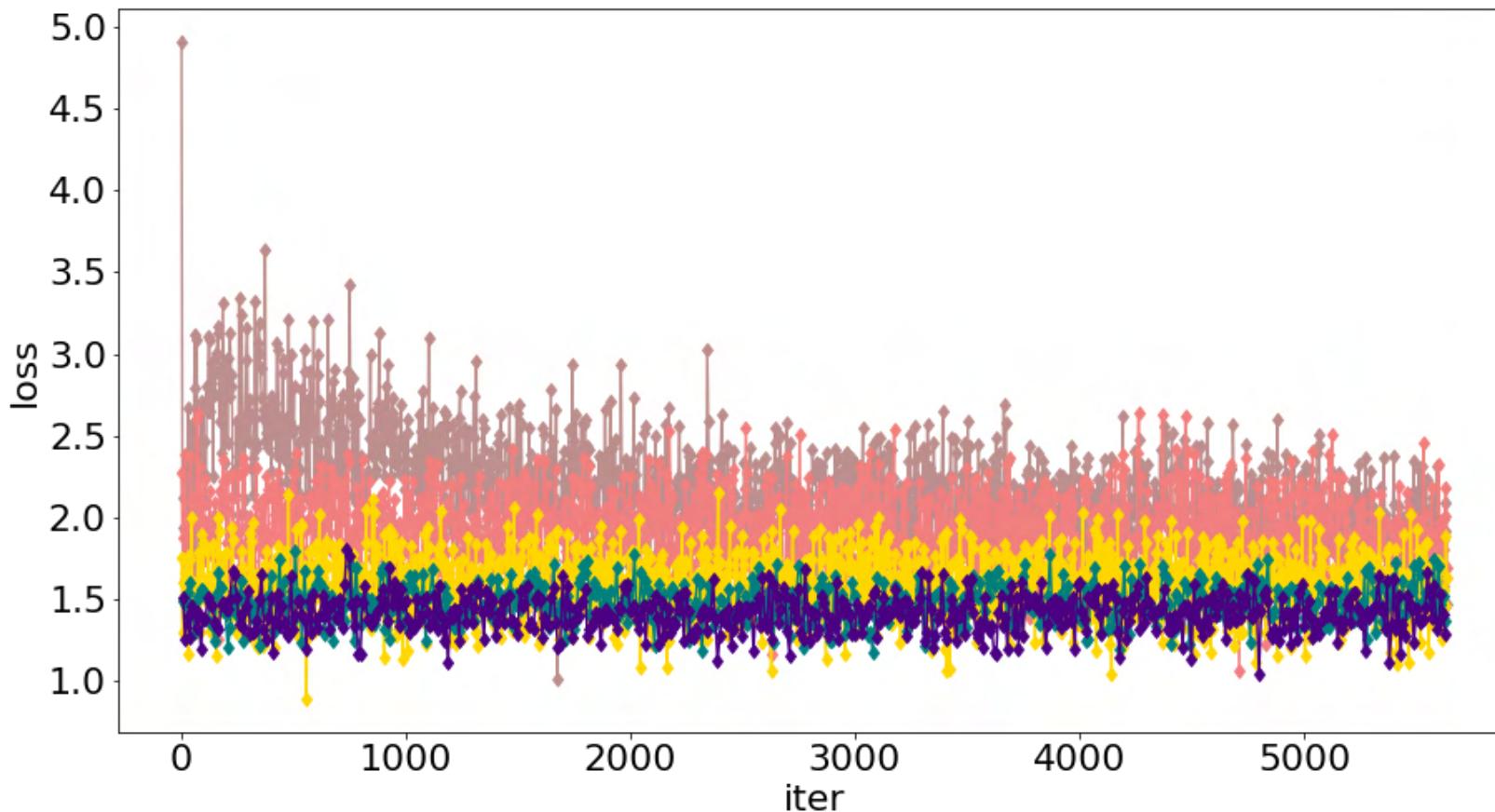
::5. 프로젝트 수행 결과

● rbbox_loss_cls 시각화



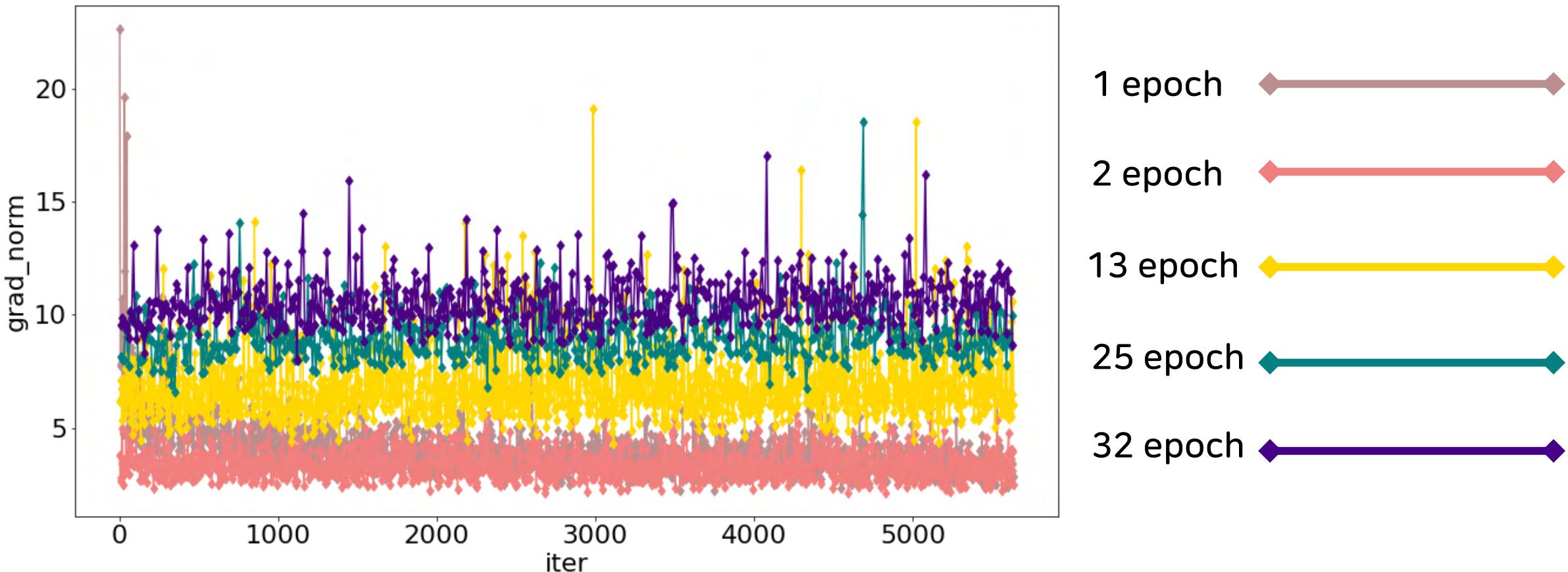
::5. 프로젝트 수행 결과

● loss 시각화



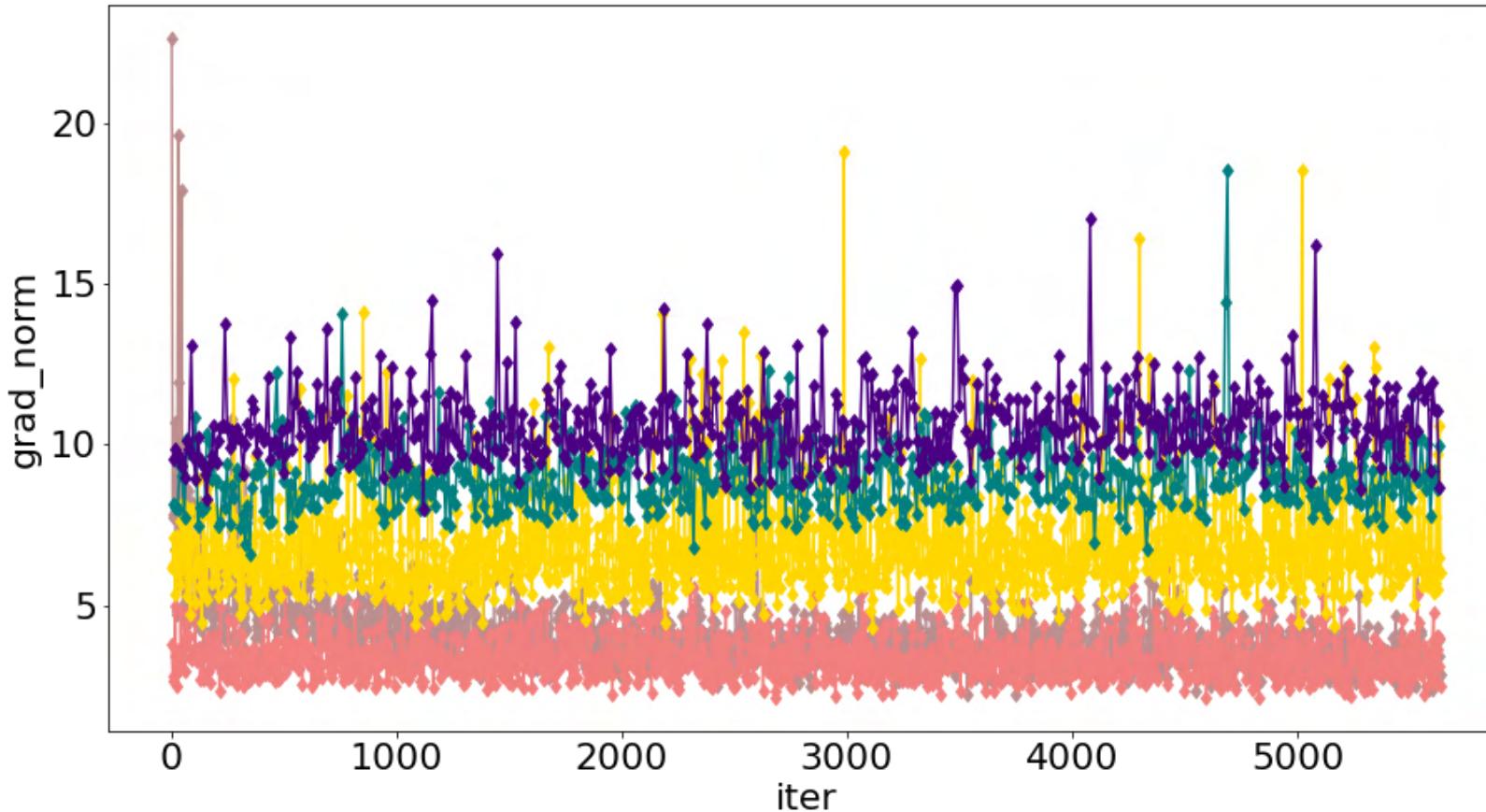
::5. 프로젝트 수행 결과

● grad_norm 시각화





grad_norm 시각화

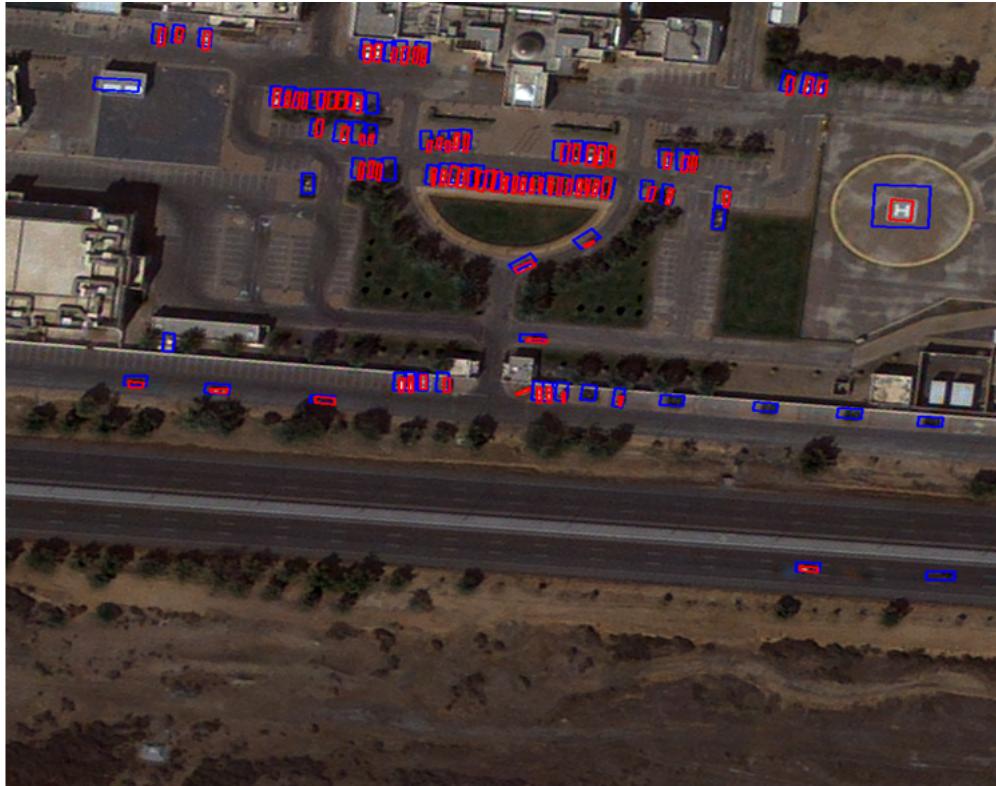


- 1 epoch
- 2 epoch
- 13 epoch
- 25 epoch
- 32 epoch

::4. 프로젝트 수행 과정

● 학습 결과 라벨링 비교

- 빨간색 박스가 **Ground Truth**, 파란색 박스가 **학습 결과**



map 평가지표를 통해서 class별 객체 검출을 확인합니다.

::4. 프로젝트 수행 과정



Crop & Merge

Crop & Merge 큰 이미지에 대해서 검출을 해야하므로 제공받은 이미지(12,362x11,344)을 효과적으로 crop/merge하여 학습하는 방법을 찾습니다.



EDA 및 전/후처리

EDA 및 전/후처리 EDA를 통해 전/후 처리를 진행하여 보다 좋은 성능을 이끌어 낼 수 있도록 합니다.

::4. 프로젝트 수행 과정

● GDAL

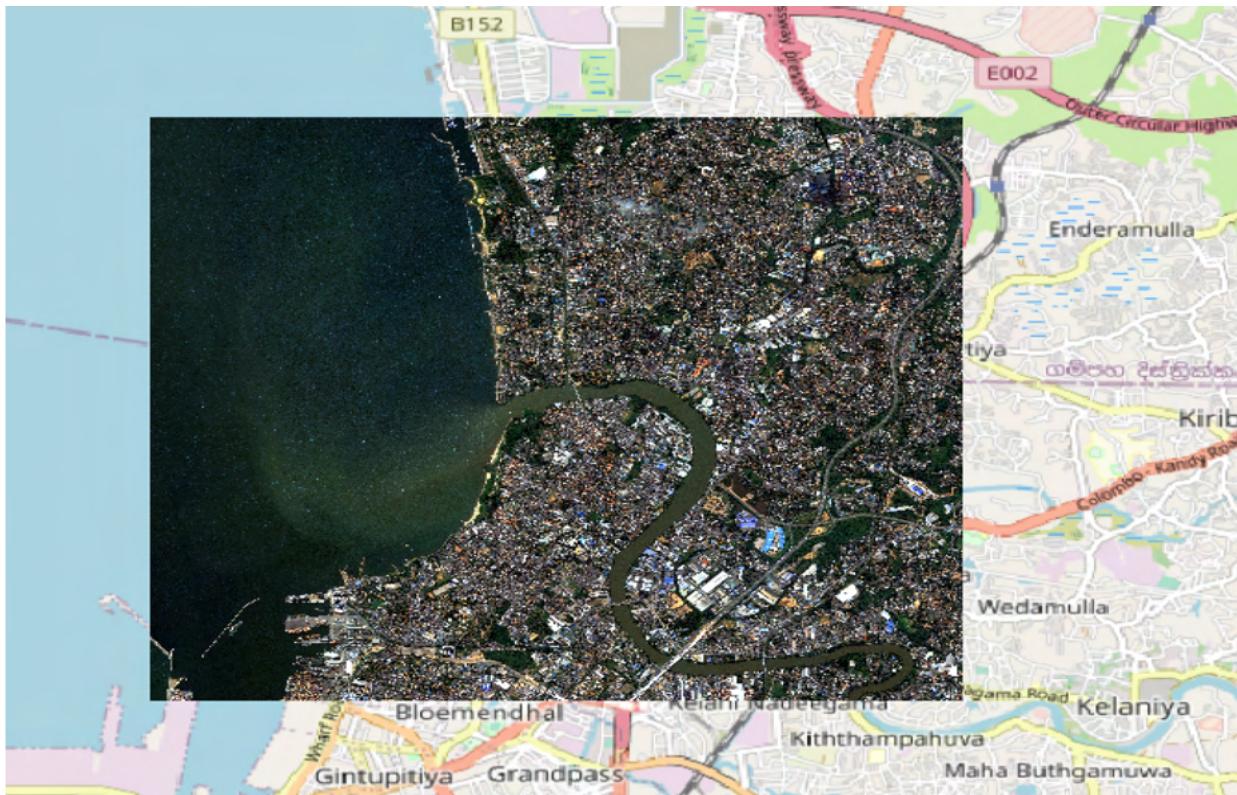
GDAL이란 광범위한 raster 포맷을
쿼리, 재투영, 왜곡(warp), 병합할 수 있는 raster 관리 도구입니다.



::4. 프로젝트 수행 과정

Crop & Merge

Gdal_translate : tif -> png , test 하기 위함
16bit 65,536 -> 8bit 256



::4. 프로젝트 수행 과정

● Step 02 - 큰 단위 이미지에서 객체검출하기



- 제공 받은 큰 이미지
 - 파일 크기 : pixel 사이즈: $(w,h) = (12362,11344)$
 - 파일 형식 : tif

사진이 크기 때문에 pil, cv2 로는 불러 올 수 없거나 정보 추출이 안될 수 있으므로 gdal 라이브러리를 사용합니다.

::4. 프로젝트 수행 과정

● 16bit tif 이미지 -> 8bit png로 변환

```
from osgeo import gdal
import numpy as np
ds = gdal.Open('modu_large.tif')
channel = np.array(ds.GetRasterBand(1).ReadAsArray())
print(channel)
```

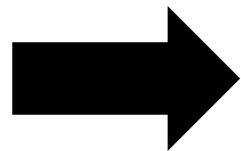
```
[[1303 1419 1341 ... 1636 1721 1848]
 [1326 1400 1376 ... 1443 1580 1786]
 [1295 1386 1405 ... 1360 1468 1651]
 ...
 [1203 1206 1235 ... 5637 5330 4743]
 [1179 1218 1286 ... 5440 5150 4874]
 [1132 1196 1243 ... 5231 5020 4902]]
```

```
from osgeo import gdal
import numpy as np
ds = gdal.Open('modu_large.png')
channel = np.array(ds.GetRasterBand(1).ReadAsArray())
print(channel)
```

```
[[20 22 20 ... 25 26 28]
 [20 21 21 ... 22 24 27]
 [20 21 21 ... 21 22 25]
 ...
 [18 18 19 ... 88 83 74]
 [18 18 20 ... 85 80 76]
 [17 18 19 ... 81 78 76]]
```

::4. 프로젝트 수행 과정

- 16bit tif 이미지 -> 8bit png로 변환



::4. 프로젝트 수행 과정

● 수행 결과

epoch에 따라 이미지를 crop 하고 객체를 검출합니다.
1,024 x 1,024에서 epoch에 따른 큰 차이는 없었습니다.

[epoch_16.pth](#)
[epoch_21.pth](#)
[epoch_32.pth](#)
[epoch_30.pth](#)

12 epoch



30 epoch



::4. 프로젝트 수행 과정

● 수행 결과

stride, 검출 규격에 따라 검출 성능이 달라짐을 확인했습니다.

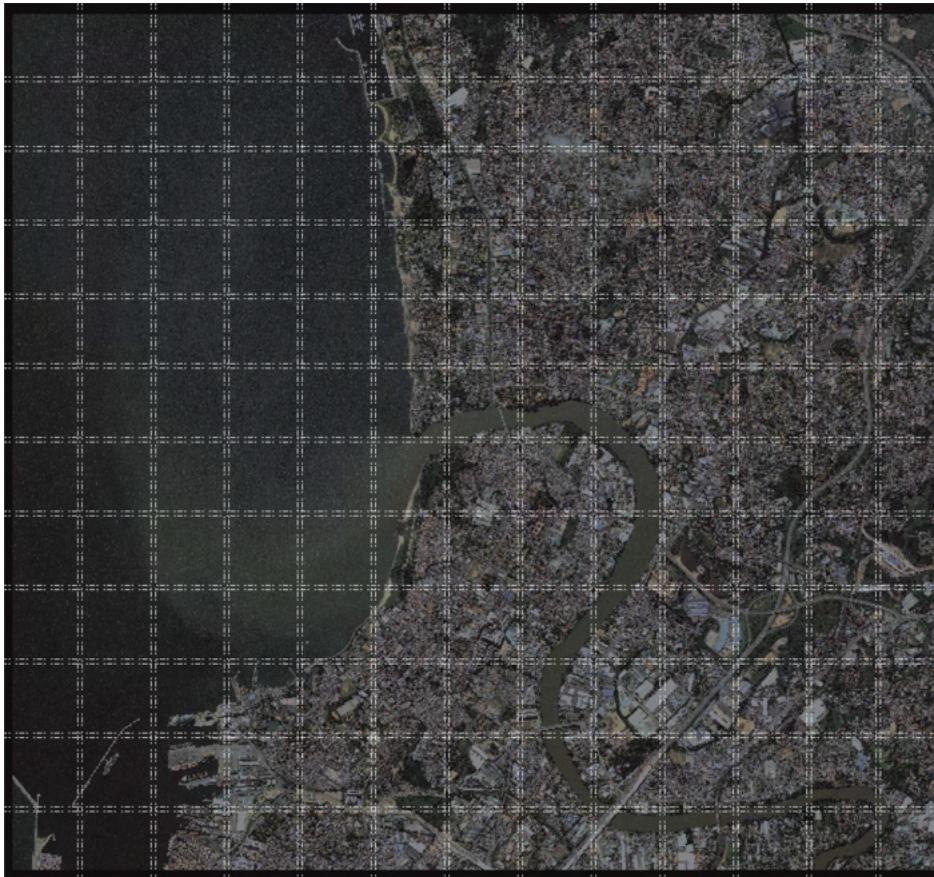
epoch = 32, 이동 크기 = 64, 검출 크기 = 512



::4. 프로젝트 수행 과정

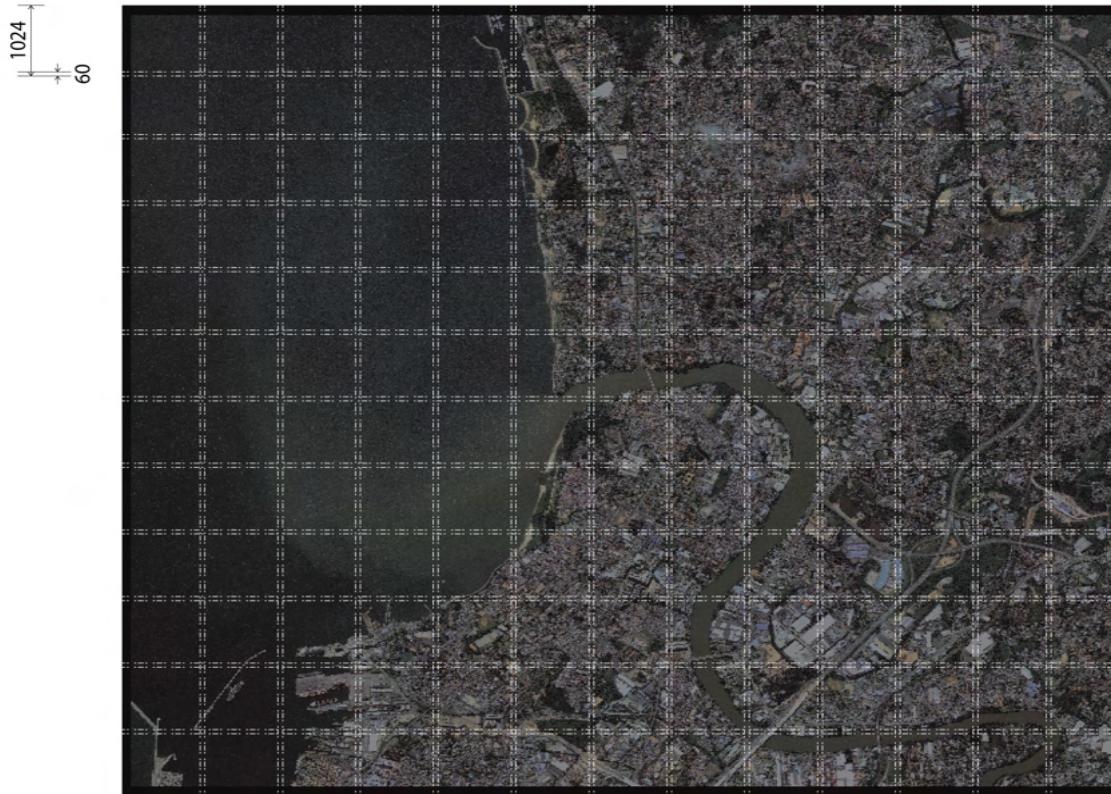
● 큰 이미지 crop 처리

경계 면에서 객체가 잘리는 경우를 대비해 padding을 하고 이미지를 일정하게 겹치도록 사진을 crop해줍니다.



::4. 프로젝트 수행 과정

● 시도한 것



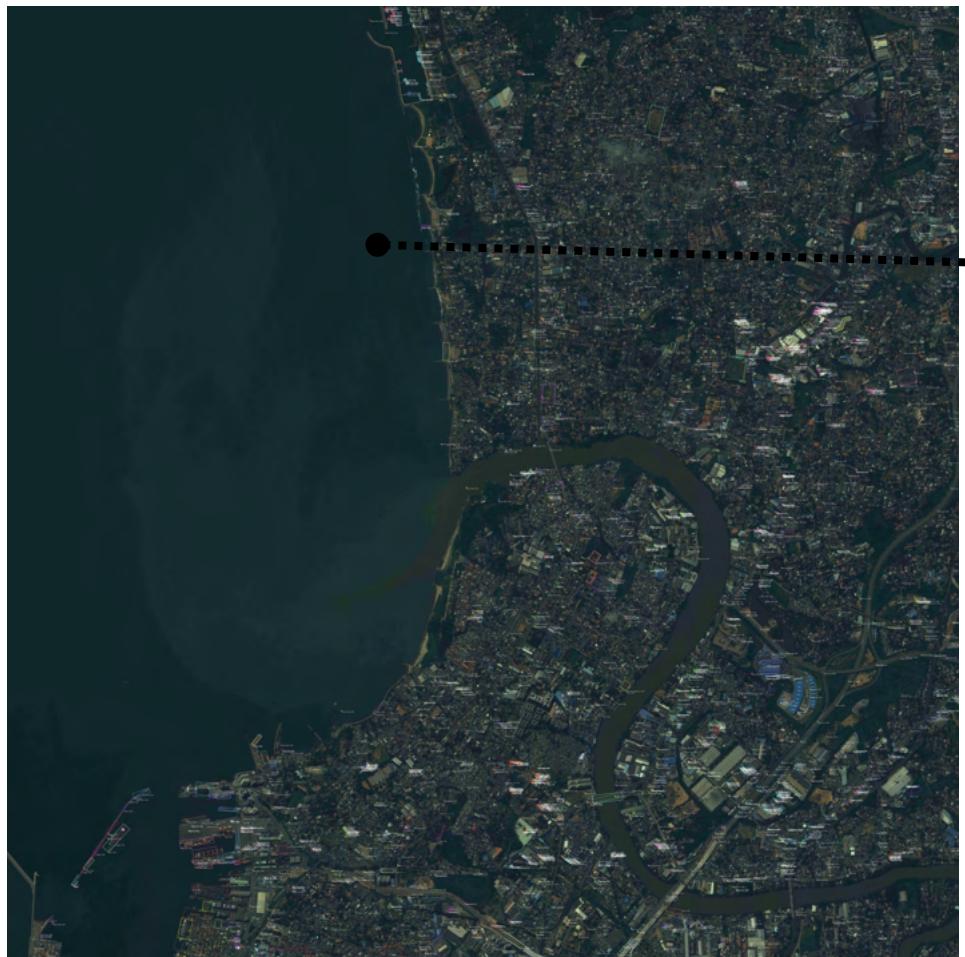
- 1,024 x 1,024 size로 crop을 할 경우 1,024 size로 딱 떨어지지 않고 겹치는 부분을 생각해서도 size가 딱 떨어지지 않아서 패딩을 한 후 crop을 하려 했으나, 패딩 처리가 좌우로만 생기는 현상이 발생했습니다.
 - 현업에서는 가장자리보다는 중심부에 필요한 부분만 활용하는 경우가 많다고 하여, 어느 정도 잘리는 부분을 감안해서 crop을 실행합니다.

05.

프로젝트 수행 결과

::5. 프로젝트 수행 결과

시도한 것



epoch = 32, 이동 크기 = 64 , 검출
크기 = 512

:: 5. 프로젝트 수행 결과

● 테스트 데이터 셋에서의 객체 검출 결과

데이터-테스트의 라벨에서 확인한 총 객체 수는 116295개.

클래스

(0-배경, 1-작은 배, 2-큰 배, 3-항공기, 4-군용항공기, 5-승용차, 6-버스,
7-트럭, 8-기차, 9-크레인, 10-다리, 11-기름통, 12-댐, 13-실내 운동장,
14-야외 운동장, 15-헬기이착륙장, 16-로터리

5. 프로젝트 수행 결과

테스트 데이터 셋에서의 객체 검출 결과

12epoch

file_name	class_id	confide	point1_x	point1_y	point2_x	point2_y	point3_x	point3_y	point4_x	point4_y
OBJ03974_	5	0.857486	37.02002	345.9178	25.30865	347.8544	21.31549	323.7068	33.02685	321.7701
OBJ03974_	5	0.409507	47.55452	344.0423	36.51976	345.8361	32.46993	320.9221	43.50469	319.1284
OBJ03974_	5	0.815416	64.0353	342.3463	53.51138	344.4636	48.73682	320.7328	59.26075	318.6155
OBJ03974_	5	0.800294	118.1807	336.7186	105.4957	338.5648	101.7667	312.9421	114.4517	311.0959
OBJ03974_	5	0.756585	137.1157	333.3761	122.8679	334.9726	120.0386	309.7235	134.2864	308.127
OBJ03974_	5	0.819647	158.3065	330.2767	144.5691	331.9452	141.3994	305.8469	155.1368	304.1785
OBJ03974_	5	0.375849	167.6345	329.6454	156.4074	330.528	154.3456	304.3019	165.5727	303.4193
OBJ03974_	5	0.387802	170.1058	587.1944	157.4903	589.129	153.7332	564.6282	166.3487	562.6937
OBJ03974_	5	0.801537	199.9637	325.1239	186.5028	326.4937	184.0591	302.4788	197.52	301.109
OBJ03974_	5	0.311343	211.2221	325.9873	200.0602	326.4274	199.0609	301.0816	210.2227	300.6414
OBJ03974_	5	0.805345	222.6251	631.5452	221.3853	619.02	248.5419	616.332	249.7816	628.8572
OBJ03974_	5	0.754276	243.0529	321.3911	229.6664	322.7382	227.2509	298.7357	240.6375	297.3886
OBJ03974_	5	0.308062	253.1407	320.3716	242.0112	321.409	239.7566	297.2209	250.886	296.1835
OBJ03974_	5	0.322624	263.69	792.9101	263.4936	803.8882	232.2066	803.3286	232.403	792.3504
OBJ03974_	5	0.484343	264.8401	319.1595	253.5584	320.1191	251.4418	295.2363	262.7235	294.2766
OBJ03974_	5	0.843272	277.3948	318.9577	265.5773	319.8377	263.9153	297.5181	275.7328	296.6381

14204개 중 21개의 레코드가 있습니다.

12epoch에서 검출된 객체는 142042

12epoch 0.3컨피던스 이상으로 검출한 객체 수 92305

3974-166 파일에서의 검출(0.3컨피던스 이상)

12epoch - 총 35(id:5,6,7,8,(10))

32epoch

file_name	class_id	confide	point1_x	point1_y	point2_x	point2_y	point3_x	point3_y	point4_x	point4_y
OBJ03974_	5	0.811602	39.85448	347.7833	26.76854	350.3033	21.80452	324.5259	34.89046	322.0059
OBJ03974_	5	0.72254	64.35176	343.471	52.70404	345.6314	48.23224	321.5219	59.87996	319.3615
OBJ03974_	5	0.860844	116.5002	339.9205	104.9791	340.7157	103.1871	314.7515	114.7082	313.9563
OBJ03974_	5	0.859327	137.941	337.3319	126.4933	337.848	125.414	313.9078	136.8616	313.3917
OBJ03974_	5	0.675574	156.4661	333.6529	145.0532	334.8701	142.2649	308.7262	153.6777	307.5089
OBJ03974_	5	0.411621	174.1521	588.957	158.0377	592.5256	152.2635	566.4519	168.3779	562.8833
OBJ03974_	5	0.752805	200.2915	326.7624	188.4409	327.3162	187.3232	303.4005	199.1738	302.8467
OBJ03974_	5	0.364613	211.4683	325.1532	199.8952	325.7751	198.5108	300.0146	210.0839	299.3926
OBJ03974_	5	0.798651	224.1876	631.6725	222.8901	617.3762	250.6289	614.8587	251.9264	629.1551
OBJ03974_	5	0.724122	241.4861	324.4764	231.2994	325.4488	229.0764	302.1611	239.263	301.1887
OBJ03974_	5	0.422909	264.0937	319.4327	253.5531	320.5379	251.0471	296.6369	261.5877	295.5317
OBJ03974_	5	0.893051	276.3193	321.9587	267.3407	322.9275	264.7981	299.3626	273.7767	298.3938
OBJ03974_	5	0.436727	284.5512	357.1292	271.2002	357.9478	269.5439	330.9334	282.8949	330.1149
OBJ03974_	5	0.926497	328.3417	351.466	316.2	352.3116	314.5621	328.7927	326.7038	327.9471
OBJ03974_	5	0.317346	336.6126	459.3812	325.8708	459.576	325.3836	432.7228	336.1254	432.5279

19956개 중 17개의 레코드가 있습니다.

32epoch에서 검출된 객체는 119956

32epoch 0.3컨피던스 이상으로 검출한 객체 수 82128

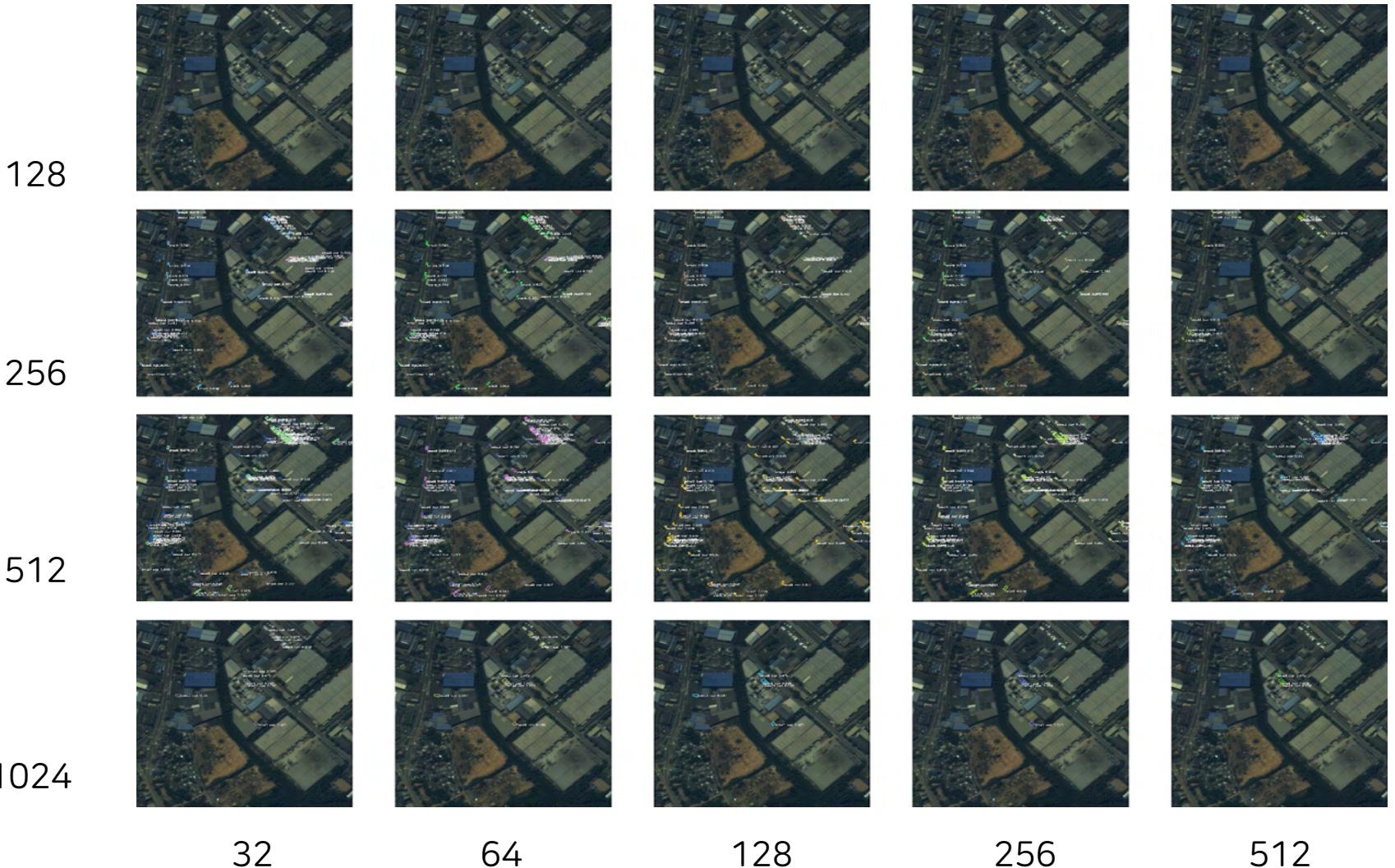
3974-166 파일에서의 검출(0.3컨피던스 이상)

32epoch - 총 42(id:5,6,7,8)

:: 5. 프로젝트 수행 결과

crop 결과 (12 epoch)

커널 크기



이동
거리

::5. 프로젝트 수행 결과

**crop 결과
(16 epoch)**

커널 크기

128



256



512



1024



32

64

128

256

512

이동
거리

::5. 프로젝트 수행 결과

**crop 결과
(21 epoch)**

커널 크기

128



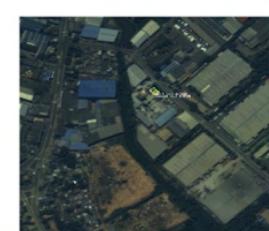
256



512



1024



32

64

128

256

512

이동
거리

::5. 프로젝트 수행 결과

crop 결과 (25 epoch)

커널 크기

128



256



512



1024



32

64

128

256

512

이동
거리

::5. 프로젝트 수행 결과

**crop 결과
(29 epoch)**

커널 크기

128



256



512



1024



32

64

128

256

512

이동
거리

06.

느낀 점

::6. 느낀 점

● 프로젝트에서 아쉬운 점

- 초반에 GCP 환경설정에 많은 시간을 투자한 점이 아쉬웠다.
- 객체 검출에 특화된 다양한 모델들을 많이 이용하지 못한 점이 아쉬웠다.
- test시 객체 검출을 count 하지 못한 점이 아쉬웠다.
- 기본 제공된 모델이 성능까지 훌륭한 모델이었기 때문에 원만하게 수행할 수 있을 거라고 생각했지만 성능이 좋은 만큼 아직은 미숙한 우리가 다루기엔 너무 복잡한 코드로 구성된 모델이었다.
- 차라리 검색을 통해 쉽게 찾을 수 있는 기본적인 모델을 이용해 최초 검출을 시도하고, 이후에 성능을 올리는 개량작업을 하는 것이 효율적으로도, 위성 영상의 객체 검출 과정을 온전히 이해하는 데도 좋았을 거라는 아쉬움이 있었다.
- 모델을 세세히 살피지 못해 실수가 있었는데, 아무래도 학습 상황이 20 epoch쯤부터 더이상 좋아지지 않았던 이유는 validation 설정이 안되어 있었기 때문이 아닌가 생각이 들었다.

::6. 느낀 점

● 프로젝트에서 수행 중 어려움 극복 사례

- 매일매일 어려웠지만 재미있게 진행하며 어려움을 극복해 나갔다.
- GCP 환경설정 어려움으로 colab과 같이 진행 했는데 둘 다 성공하여 원활한 진행을 할 수 있었다.
- 대형 이미지의 처리에 있어, 패치를 준 후 crop을 진행하는 방향으로 회의를 통해 적절한 방법을 도출했다.
- crop된 이미지에서 test시 적절한 parameter를 알지 못했으나, stride와 Kernel 크기를 조정하여 수치 별 총 검출량과 오검출량을 비교해보며 최적의 값을 찾으려 했다.
- train시 1epoch당 4시간 이상 걸리는 문제가 있어 checkpoint를 설정하여 시간을 절약했다.

감사합니다.